PHIL 7001: Fundamentals of AI, Data, and Algorithms

Week 9
Introduction to Neural Networks and Deep Learning

Maomei Wang
Email: mmmw@connect.hku.hk

Introduction

Contents

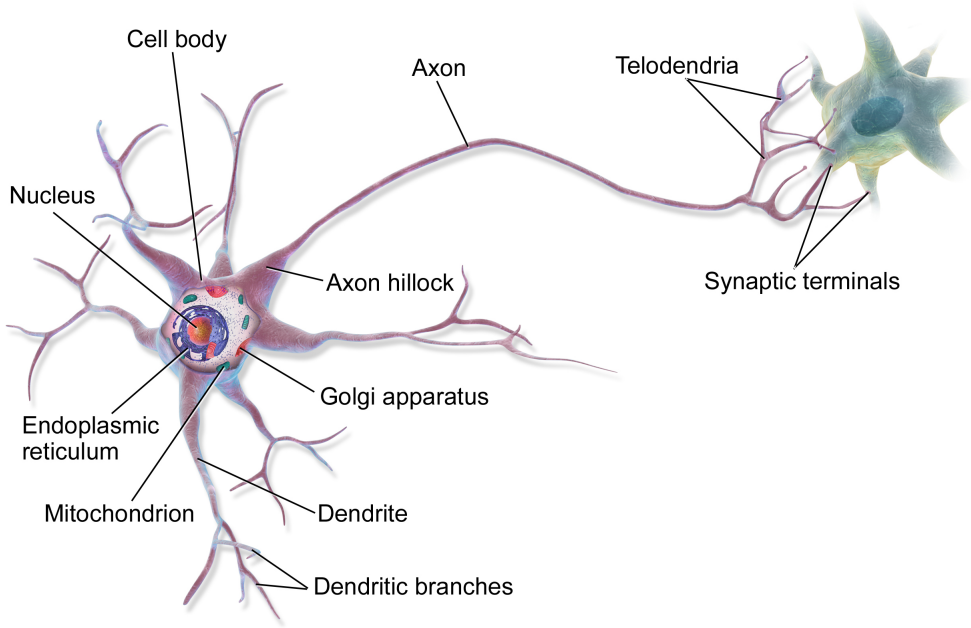
- Introduction to Artificial Neural Networks (ANNs):
  - From neurons in the brain to ANNs
  - Function Approximation
- Deep Learning (DL):
  - Machine Learning (ML) basics: recap
  - ML vs. DL
  - Why is DL important
  - Employing DNNs as models in DL
  - Learning DNNs
  - History, applications, and problems

Notice

- You will see a lot of terms, e.g., ANN, DNN, DL, SDG, etc. Please forgive me for this; being familiar with these terms will be helpful if you want to further study related issues.
- You will see a lot of mathematical formalism, which may frustrate some of you. Although I need to present the math involved, for math is a way to make things precise, you are not required to understand the math at a high level. Please bear in mind that having a conceptual understanding of the related things is enough for this course (and for exam).
- Please email me regarding any difficulties you have encountered during your studies. Let's collaborate to find solutions for these challenges.

Artificial Neural Networks

Neurons in the brain

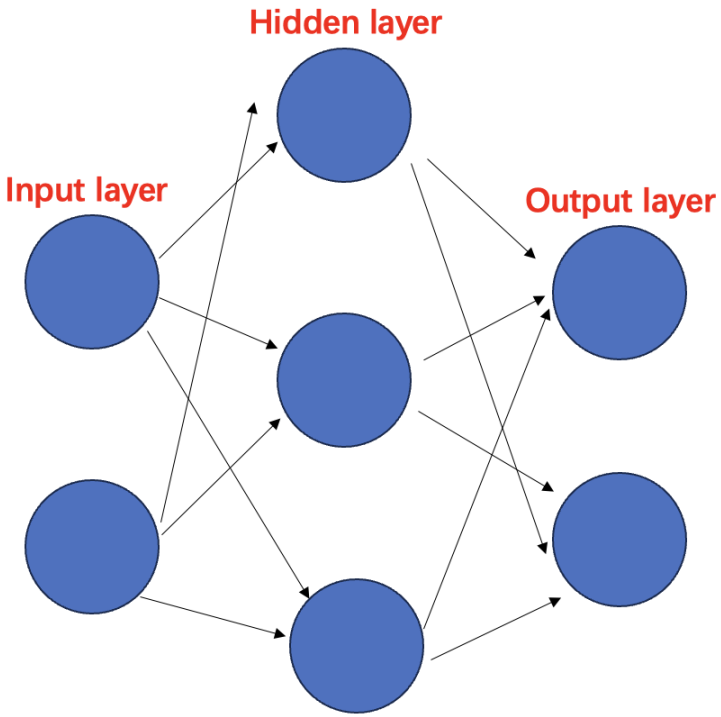


- Here is a video illustration: `https://www.youtube.com/watch?v=hGDvvUNU-cw`.

ANNs are inspired by the human brain

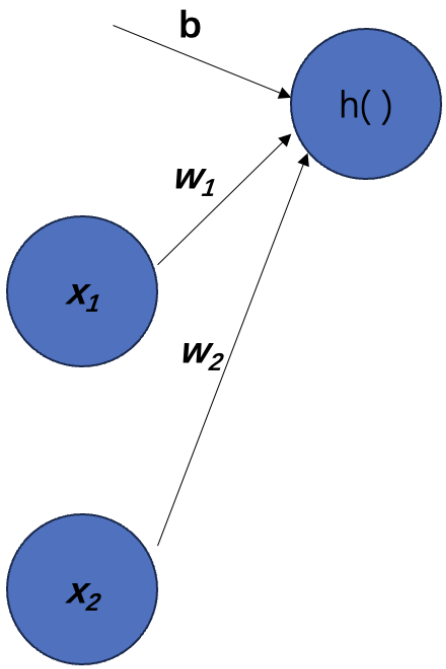
- Each ANN consists of layers.
- Layers consist of nodes, which we will call units, that receive stimuli and transform the stimuli and output a stimulus;
- These units can be trained only to pass forward useful signals.

An example



- This is a two-layer (fully connected) ANN;
- In our real-life applications, ANNs can have a large number of hidden layers and will be called Deep Neural Networks (DNNs).

For the hidden layers



- **Weighted sum:** For each unit, we get the weighted sum plus a bias: $z = x_1 w_1 + x_2 w_2 + b$, where w_1, w_2 are weights, and b is a bias.
- **Using an activation function to do transformation:** For each unit, it transforms the weighted sum using a function called activation function, denoted as h, and then output $h(z)$.
- The outputs of one layer are the inputs to the next.

Example activation functions

- **sigmoid function**

$$\text{sigmoid}(x) = \frac{1}{1+\exp(-x)} \tag{1}$$

- **ReLU**

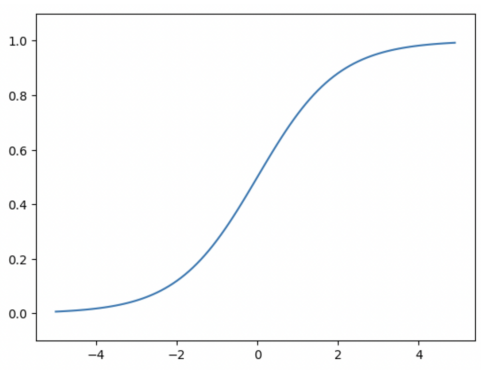
$$\text{ReLu}(x) = \begin{cases} x, & x > 0, \\ 0, & x \leq 0. \end{cases} \tag{2}$$

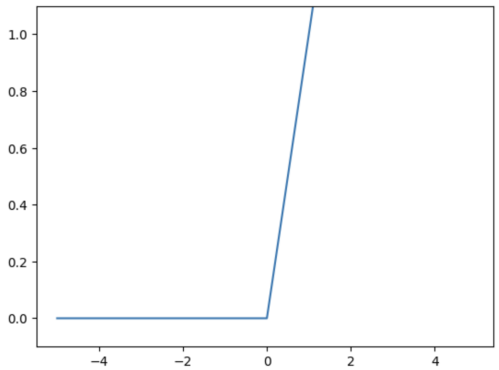
- **softplus**

$$\text{softplus}(x) = \log(1+e^x). \tag{3}$$

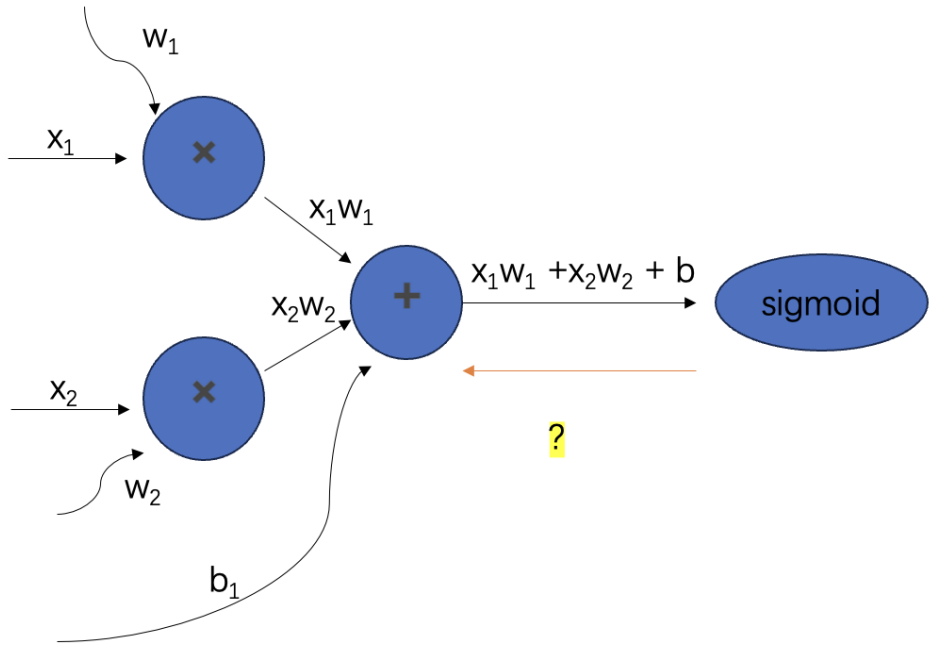
- **tanh**

$$\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}. \tag{4}$$



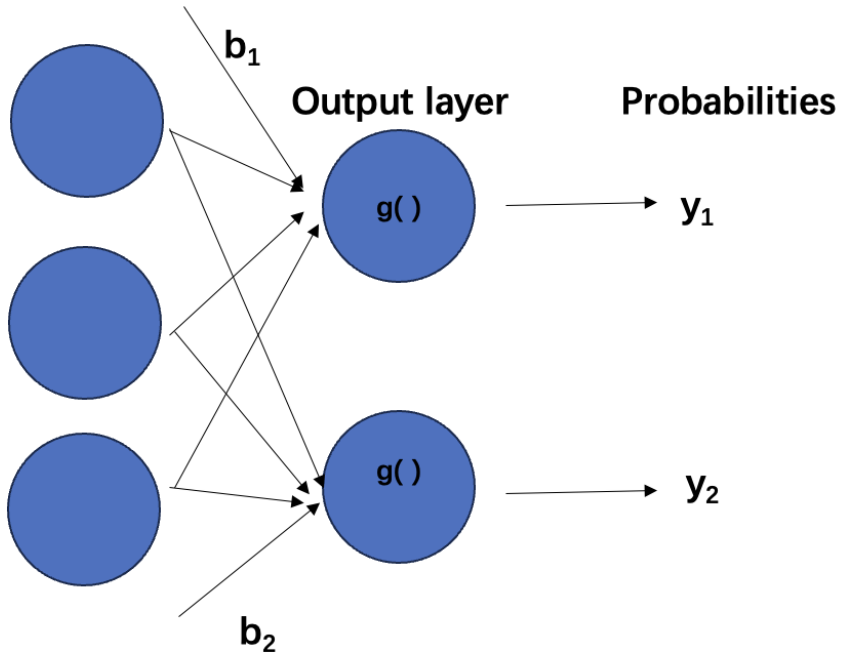


Computational graph



- Another way of considering the above process is depicted by the computational graph, which will be relevant for learning.

The output layer



- **Weighted sum:** For each unit, we get the weighted sum: $z = w_1 h(z_1) + w_2 h(z_2) + w_3 h(z_3) + b$, where $h(z_1), h(z_2), h(z_3)$ are transformed signals, w_1, w_2, w_3 are weights, and b is a bias.
- **Using the softmax function to do transformation:** For each unit, it transforms the weighted sum using an activation function, denoted as g, and then outputs $g(z)$.

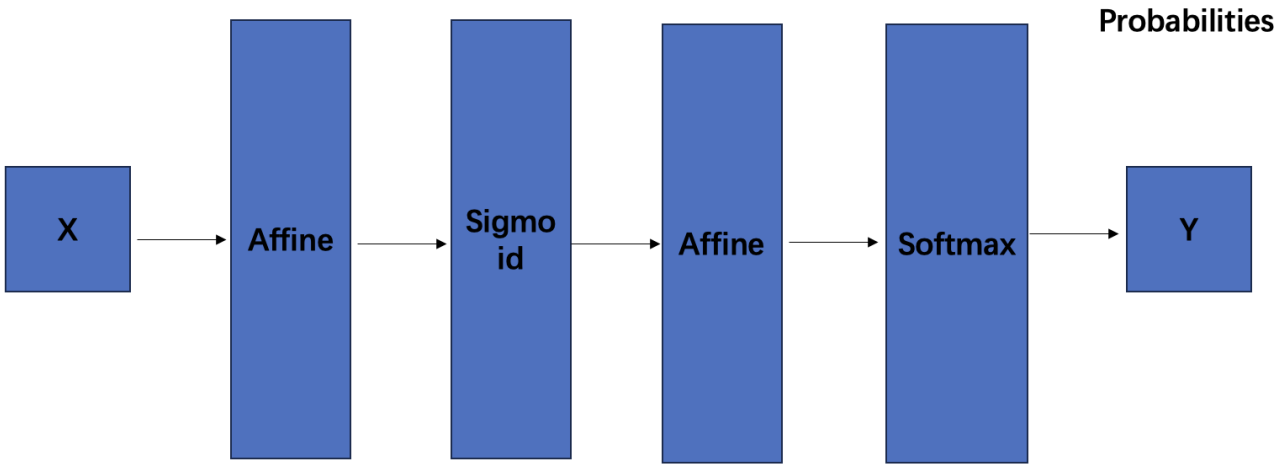
Softmax function

- The softmax function:

$$\text{softmax}(z_k) = \frac{\exp(z_k)}{\sum_{i=1}^{n} \exp(z_i)} \tag{5}$$

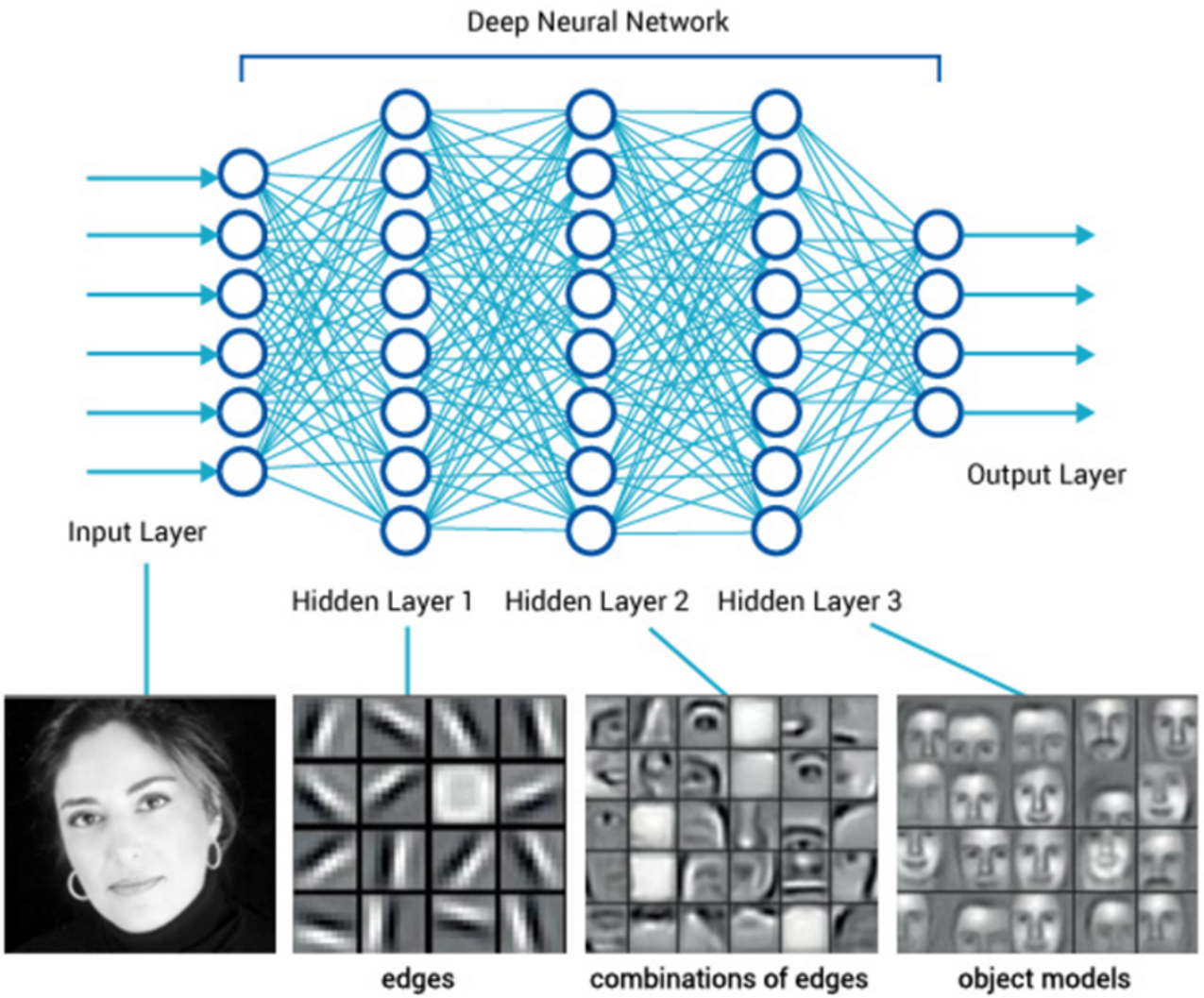
  where n is the number of units in the output layer. So this function, in fact, turns a vector of n real values into a vector of n real values that sum to 1.
- Clearly, by using the softmax function to do transformation, each output is a number in [0,1], and they satisfy Addivitiy — these make them look like probabilities.
- But sometimes we employ the identity function.

Summary so far



- The above process from an input **x** to output is called forward propagation. This kind of neural network is also called Feedforward Neural Network (FNN) (or Multilayer perceptron (MLP)), for the information flows from the input to the output with no circles or loops. Later, you will meet two other kinds of neural networks, called CNN and RNN. These are the three main kinds of neural models.

FNN


Deep Neural Network
Input Layer
Hidden Layer 1
Hidden Layer 2
Hidden Layer 3
Output Layer
edges
combinations of edges
object models

DNNs as functions

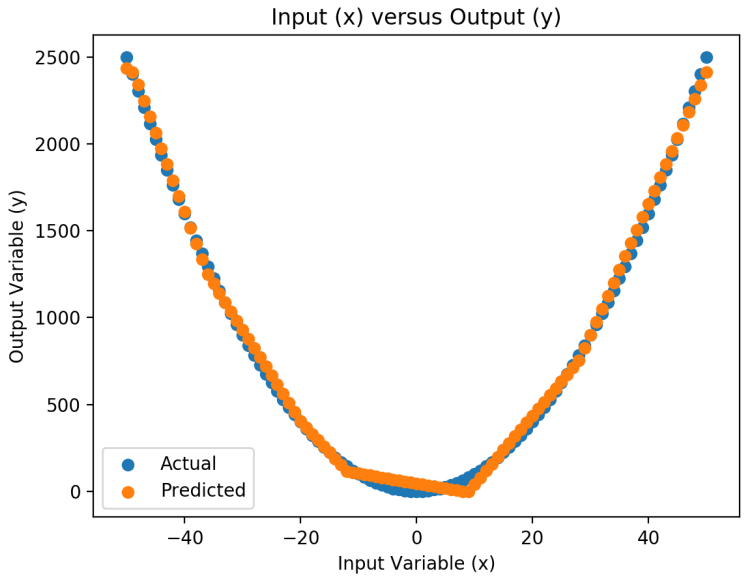
- The resemblance to real neural cells and structures is superficial. And it is better to understand it from a mathematical point of view.
- For our case, let $f^{(1)}$ denote the transformation function in the first layer and $f^{(2)}$ the transformation function in the second layer. Then clearly we have $f(\mathbf{x}) = f^{(2)}(f^{(1)}(\mathbf{x}))$ for each $\mathbf{x} \in \mathbf{X}$, which consists of $f^{(1)}$ and $f^{(2)}$. We may ignore the bias and further specify it as

$$f(\mathbf{x}) = h^{(2)}(\mathbf{W}^{(2)} h^{(1)}(\mathbf{W}^{(1)}\mathbf{x})), \tag{6}$$

where $\mathbf{W}^{(i)}(i \in \{1, 2\})$ denotes the weights for the i_{th} layer.

Function approximation

Universal approximation theorem: Any continuous function $f : \mathbb{R}^m \to \mathbb{R}^n$ Can be realized by a network with one hidden layer (given enough hidden neurons).



- In English, a network with just two layers of computational units, the first nonlinear and the second linear, can approximate any continuous function to an arbitrary degree of accuracy. So, this model does have great expressive power (but may overfit).

Architecture

In designing an ANN, we may consider

1. the number of hidden layers (depth)
2. the number of units for each hidden layer (width).

During practice, since deeper NNs normally behave better, we employ DNNs more frequently, i.e., we employ DNNs that can be expressed as, e.g.,

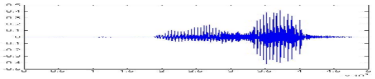
$$f(\mathbf{x}) = g(\mathbf{W}^{(n+1)} h^{(n)}(\mathbf{W}^{(n)} \dots h^{(3)}(\mathbf{W}^{(3)} h^{(2)}(\mathbf{W}^{(2)} h^{(1)}(\mathbf{W}^{(1)} \mathbf{x}))))) \tag{7}$$

(For deep FNNs, but others are similar.)

Deep Learning

DL examples

- Speech Recognition

$f($  $)=$ "How are you"

- Image Recognition

$f($  $)=$ "Cat"

- Playing Go

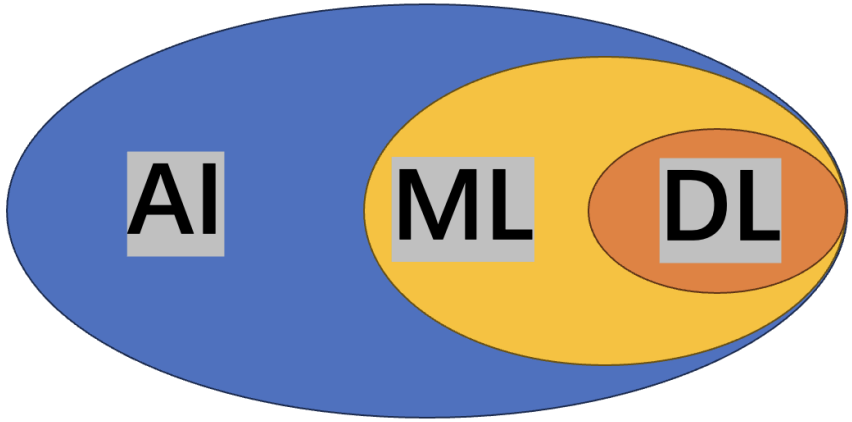
$f($  $)=$ "5-5" (next move)

- Dialogue System

$f($ "Hi" (what the user said) $)=$ "Hello" (system response)

Wild applications!

DL is a special kind of ML, so we will begin with ML.



Machine Learning (ML): Recap I (general idea)

A very short answer to the question of what ML is goes as follows:

- An ML process is a process to approximate some function
- The function that the machine is looking for is assumed to be the true function.

Notice that one underlying assumption is that there exits some underlying true function, and the function does not change (this does not mean that the function should be static).

> What if there does not exist any true underlying unchanging function?

Machine Learning (ML): Recap II (important components a)

- **Data.** Supervised (labeled data) vs. Unsupervised (unlabeled data) vs. Reinforcement learning (only labeled with reward).
- **Model.** A model means a form of a function in what follows. Normally, we do not know which is the "true" function, so we begin with a set of possible (we think) functions, and this set is called hypothesis space. In this course, you have already met several models, such as Linear model, logistic model, etc. Formally, a hypothesis space is:

$$\mathcal{F} = \{f(\mathbf{x};\theta)|\theta \in \mathbb{R}^n\}$$

$f(\mathbf{x};\theta)$ is a function with θ as its parameters, n is the number of the parameters.

Machine Learning (ML): Recap II (important components b)

- **Loss function.**
  Let L denote a loss function that maps a pair of $(y, f(\mathbf{x}))$ to a number, such that $L(y, f(\mathbf{x}))$ represents the distance between y and $f(\mathbf{x})$. Intuitively, the bigger $L(y, f(\mathbf{x}))$ is, the further y is from $f(\mathbf{x})$, thus the more inaccurate the prediction made by f is.
  - Mean squared error.
  - Cross entropy function.

Why are these loss functions reasonable measures of inaccuracy or distance between two functions?

Machine Learning (ML): Recap II (important components c)

Let the empirical risk for N data be denoted as R and defined as:

$$R(f) = \frac{1}{N} \sum_{i=1}^{N} L(y_1, f(\mathbf{x}_1)).$$

- **Learning strategy. ERM (Empirical Risk Minimization)** — minimizes the empirical risk:
$$f^\star = \min_{f \in \mathcal{H}} R(f).$$
- **Optimization algorithm.** An optimization algorithm is an iterative, step-by-step procedure for computation. When you have chosen your evaluation strategy, this is where the machine helps you find the optimal one. (Due to computational capacities, normally, machines will find something they believe to be the closest, which may not really be the optimal one.) Example: Gradient Descent (GD) and its variants, e.g., Stochastic Gradient Descent (SGD).
- **Evaluation of the model's performance.** We may use accuracy rate/error rate, precision rate/recall rate, etc., to evaluate the model.

Why are these learning strategies reasonable?

Feature engineering in classical ML

Feature engineering: transform raw data into meaningful features.

- It is essential in classical ML. For each observation, we represent it by its feature vectors, and our functions are defined on these.
- It is labor-intensive and time-consuming, including feature selection, creation, and transformation
- It requires domain knowledge and a deep understanding of the data

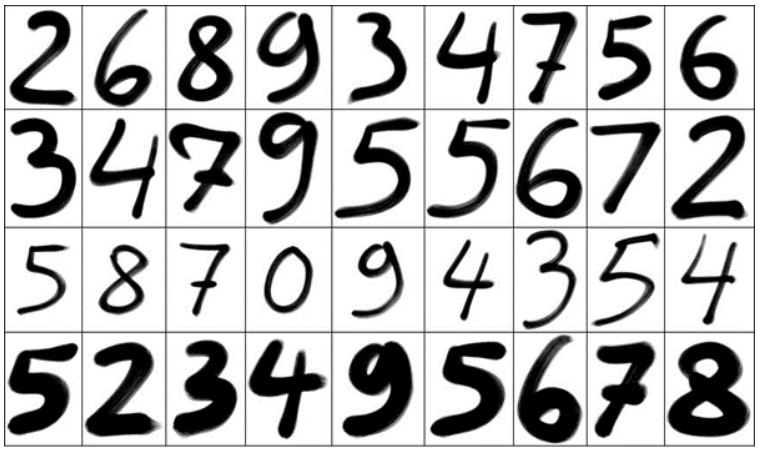
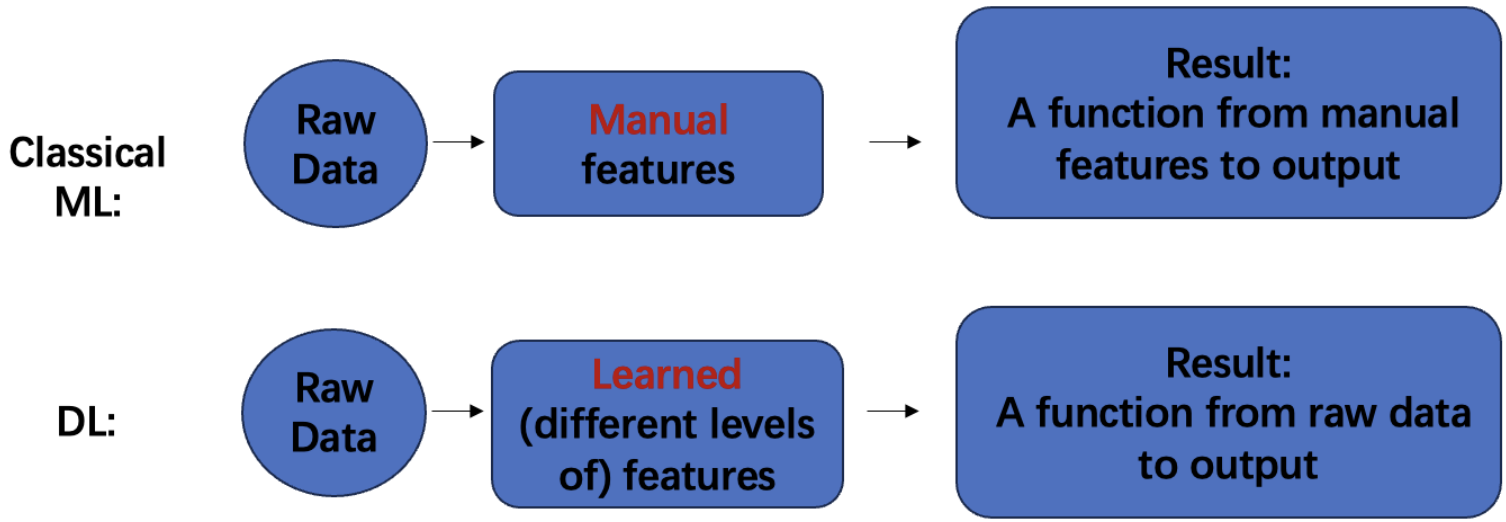


Figure: Handwritten digits

What if we are unable to offer the features?

DL vs. ML

DL is a special kind of ML that also does representation learning/feature extraction.



Feature engineering vs. Representation learning. Requiring minimal domain knowledge!
It is also called End-to-End learning.

The importance

> *The goal of AI is to* **develop** *machines that* **behave** *as though they were* **intelligent**. *(John McCarthy)*

- **Knowledge representation**: to store what it knows or hears;
- **Automate reasoning**: to answer questions and to draw new conclusions;
- **Machine Learning**: to adapt to new circumstances and to detect and extrapolate patterns;
- **Natural language processing (NLP)**: to communicate successfully in a human language;
- **Computer vision (and speech recognition)**: to perceive the world
- **Robotics**: to manipulate objects and move about.

We cannot implant all the knowledge; letting machines learn is a solution.
We cannot implant all the features; letting machines learn is also a solution.

Employing DNNs as models

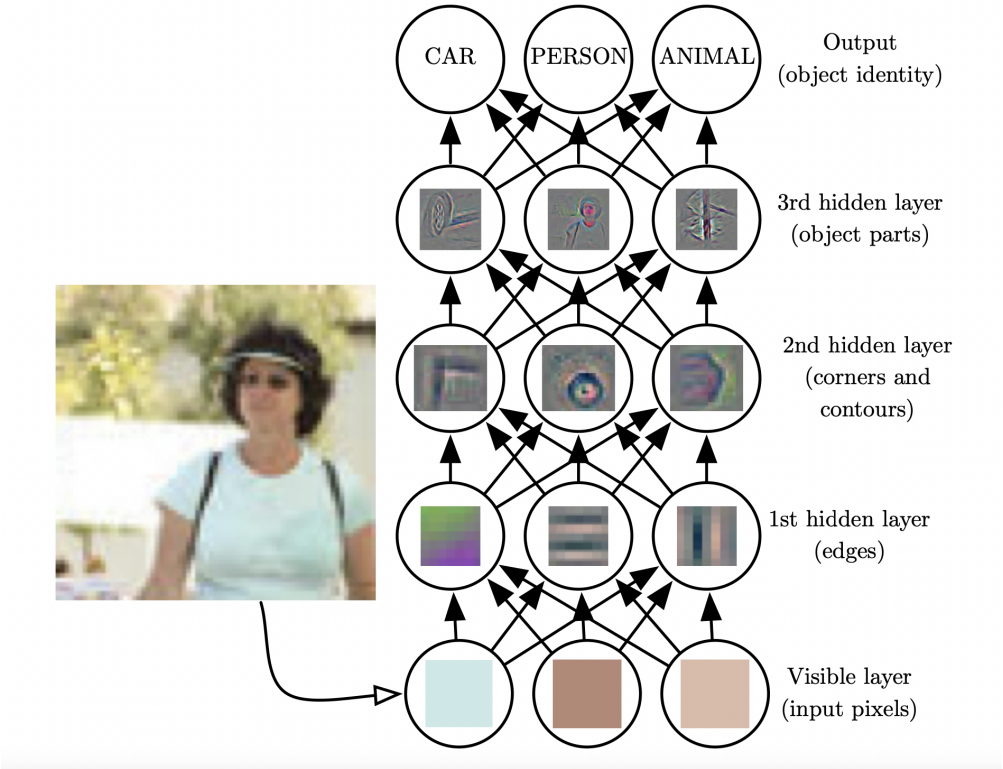
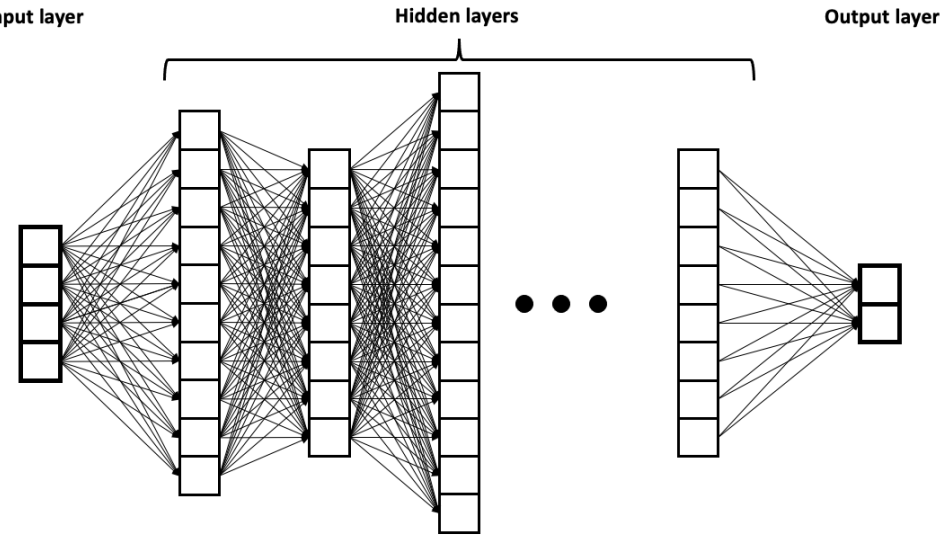


Figure: Multiple layers learn different levels of features

- Expressive power (learn non-linear functions). Thus the term "deep learning": depth of the network.
- Available training algorithms; big data; computational capacity; etc.
- But there are also other models for DL, such as Deep Belief Networks.

Deeper NNs

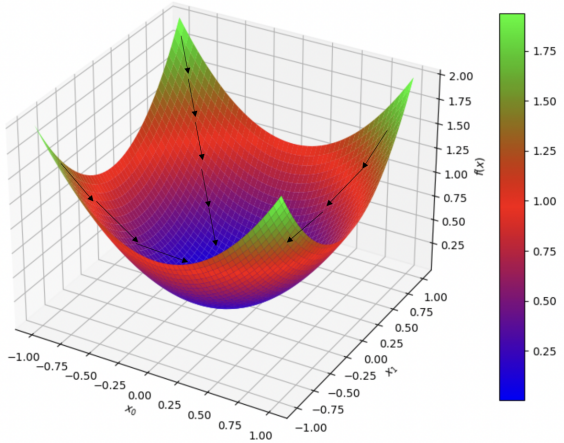
- AlexNet (2012) 8 layers
- VGG (2014) 19 layers
- GoogleNet (2014) 22 layers
- Residual Net (2015) 152 layers
- deeper and deeper?
  - Easier to train;
  - Better behavior;
  - Belief in the simplicity of the underlying mechanism, etc.

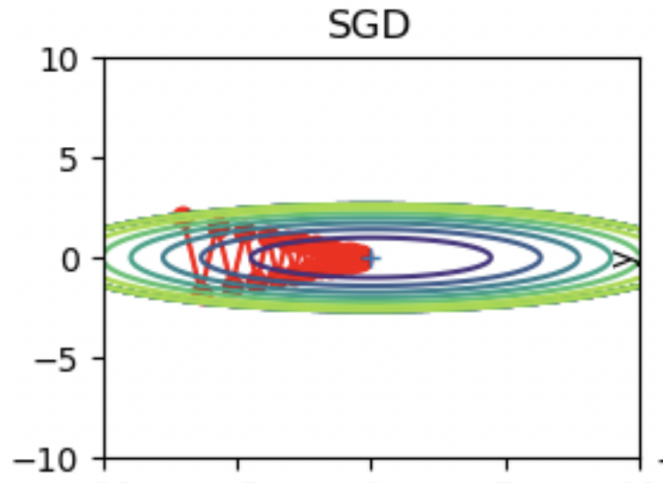


Paper: Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training very deep networks. *Advances in neural information processing systems, 28.*

Learning parameters: SGD

- Initialize parameters, i.e., weights and bias, and let the machine learn DNNs' parameters by some optimization algorithm.
- SGD is a popular optimization algorithm.





$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta} \tag{8}$$

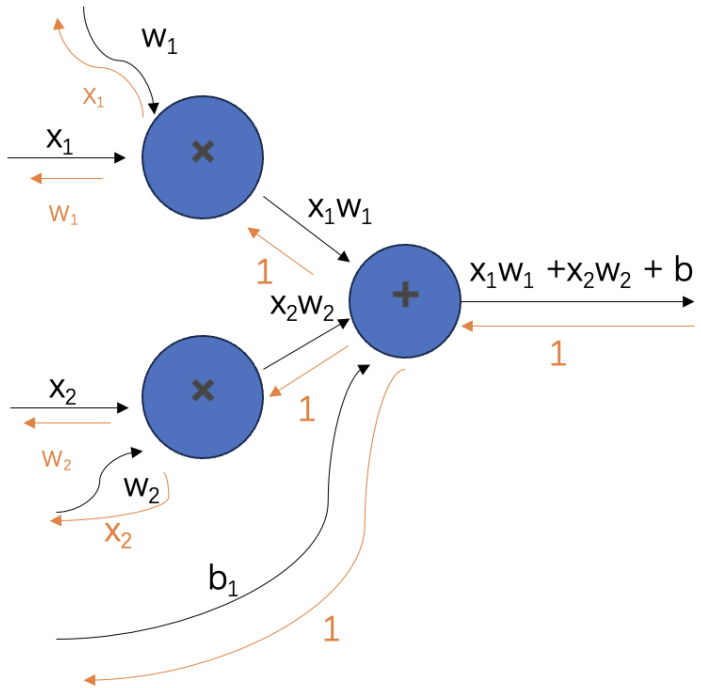
- By updating the model parameters based on gradients, SGD iteratively adjusts the model to minimize the loss.
- Repeat until converge.
- But how can we get the gradients?

Computing gradients: Backpropagation (BP)

- Chain rule: let $f = g(g'(x))$ such that $y = g'(x)$ and $z = f(x)$, we have

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} \tag{9}$$

  So for composite functions, we can easily get the gradients by calculating the gradients of their compositions.
- BP helps compute the gradients of the compositions when employing DNNs.
- General way: Automatic Differentiation (AD)



Hyper-parameters: some are based on practical wisdom

- However, setting and tuning some hyper-parameters involved in real-life practice are based on practical wisdom.
- For example, concerning setting the number of layers, no formula or theoretical result tells us the number of layers or the units per layer to use in the model.
- Similar things happen to the choices of activation function, loss function, learning rate, etc.
- When building a model, scientists have to do several trial runs with different values for these hyper-parameters before settling on the best ones.

Data: quality and quantity

- "Data is the King."
- As our models for DL are getting larger and larger (e.g., GPT-3 has 175 billion parameters), the data we need to train them is growing, too.
- "Garbage in, garbage out."
- Toxic information.

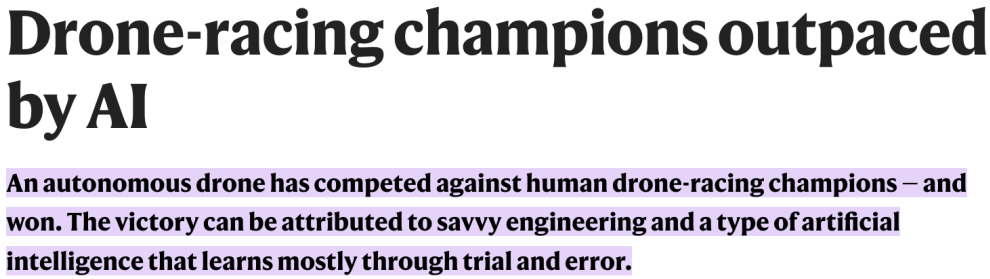
Question: Will we always have enough data? What if not? How can we have good-quality data? Btw, how to define good quality?

Milestones

- (1943) Warren McCulloch and Walter Pitts purported a simple artificial neural network model called MP Neuron Model;
- (1951) Marvin Minsky built the first neural network computer SNARC.
- (1950) Alan Turing's article "Computing Machinery and Intelligence." Therein, he introduced the Turing test, machine learning, genetic algorithms, and reinforcement learning.
- (1956) Dartmouth Conference. (Here the name Artificial Intelligence was first introduced.)
- (1962) The perceptron convergence theorem (Block et al., 1962) says that the learning algorithm can adjust the connection strengths of a perceptron to match any input data, provided such a match exists.
- (1974) Werbos invented the backpropagation algorithm.
- (1986 to present) Rumelhart, Hinton and Sejnowski and others lead to a renaissance of neural networks.
- (2001 to present) Big data.
- (2011 to present) Deep Learning.

Applications

- Healthcare and Medical Imaging: LYNA system achieves 99.6% overall accuracy in diagnosing metastatic breast cancer, which is better than an unaided human expert.
- Recommendation Systems: Amazon, Facebook, Netflix, Spotify, YouTube, Walmart, and Little Red Book (Xiaohongshu)'s excellent recommendation systems.
- Robotics and Autonomous Systems: see one recent news below



Drone-racing champions outpaced by AI

An autonomous drone has competed against human drone-racing champions – and won. The victory can be attributed to savvy engineering and a type of artificial intelligence that learns mostly through trial and error.

- Gaming and Virtual Reality: AlphaGo (surpassed all human players)

Questions: Give examples of DL products that you know. If you are unsure whether one has employed DL techniques, google it.

The problem of Explainability

- **The problem:** The challenge of understanding and interpreting the decisions or predictions made by DNNs.
- **Reason:** Deep architectures and the massive number of parameters involved.
- **Reaction:** This lack of transparency raises concerns regarding bias, fairness, and trust in AI systems. Researchers are actively developing techniques to enhance the explainability of deep learning models, enabling users to understand and trust the decisions made by AI systems.

Question: Will you trust an AI doctor if you and even the person who creates the "doctor" does not know how it has arrived at a decision?

Other problems

- Data: data bias, data storage, and privacy;
- Misspecified goals (alignment);
- Hyper-parameter tuning;
- etc.

Main points

1. DNNs are mathematical models that have good expressive power by doing nonlinear transformations. Important concepts involved: layer, node, weights, bias, and activation function.
2. DL is a kind of ML that also learns representations/features by machines.
3. DNNs are frequently employed models for DL (but not the only kind).
4. During DL, machines update parameters by algorithms such as SGD and BP. Also, we need a large amount of good-quality data and practical wisdom.
5. DL is prevailing: many real-life applications are using DL.
6. However, there are worries concerning the applications of DL in the real world, e.g., explainability, privacy, alignment, etc.

Let's have a try: Tensorflow playground

Link: `https://playground.tensorflow.org/`.

References

- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature, 521*(7553), 436-444.
- Lee Hung Yee' lecture notes `http://speech.ee.ntu.edu.tw/~tlkagk/slide/Tutorial_HYLee_Deep.pptx`
- Qiu, X. P. (2020). Neural Networks and Deep Learning. Retrieved from `https://nndl.github.io/` (Another book for practice. Link: `https://github.com/nndl/practice-in-paddle/tree/main`)
- Russell, S. J., & Norvig, P. (2022). Artificial Intelligence: A Modern Approach, 4th, Global ed.
- Also, this link `https://paperswithcode.com/` provides papers with codes.